Escuela Politécnica Superior

Sistema de autenticación robusto

Máster Universitario en Ciberseguridad

Trabajo Fin de Máster

Autor:
Iris Mª Alfonso Pagán

Tutor:
Dr. Antonio Zamora Gómez

Junio 2019

Universitat d'Alacant
Universidad de Alicante

Mis mejores agradecimientos:

A mi tutor, Antonio,
por haberme ayudado y guiado
durante el desarrollo de este proyecto.

A mi familia, por estar
siempre animándome.

Y, en especial, a mi pareja por
ser siempre mi apoyo incondicional.

Índice

Índice de ilustraciones

1. Introducción

Hoy en día, para acceder a cualquier servicio en Internet, se hace uso de un usuario y una contraseña. Este tipo de autenticación mediante credenciales es el más empleado por los usuarios, ya sea para acceder a la cuenta del banco, como para acceder a las redes sociales, correos, etc. De este modo, mediante el uso de credenciales se verifica que el usuario es quien dice ser y no un suplantador de identidad. Sin embargo, a pesar de ser la opción principal elegida entre los usuarios, este sistema posee muchas vulnerabilidades. Un usuario suele emplear la misma contraseña para todos los servicios, además, para recordarla fácilmente esta suele ser corta, y por lo tanto, débil. El desconocimiento sobre la facilidad con la que un atacante puede obtener una contraseña provoca que casi el 90% [1] de las contraseñas de los usuarios sean vulnerables.

1.1 Ataques a contraseñas

Los ciberdelincuentes están en continuo ataque para obtener las credenciales de los usuarios y robar tanto sus datos privados almacenados como obtener acceso a sus cuentas de servicios. El uso de contraseñas fáciles de recordar o la mala práctica de no cambiarlas regularmente ponen en peligro la seguridad y privacidad. Los diferentes tipos de ataques que se pueden realizar son los siguientes:

1.1.1 Fuerza bruta

Los ataques de fuerza bruta consisten en obtener la contraseña mediante prueba y error realizando distintas combinaciones. Este ataque es muy costoso y con contraseñas robustas (largas y complejas) resulta inviable, no obstante, con una capacidad de cómputo adecuada y un tiempo relativamente largo, es teóricamente viable este ataque.

1.1.2 Diccionario

Este tipo de ataque consiste en probar todas las palabras existentes en un diccionario compuesto de las contraseñas más usadas. Estas contraseñas proceden de fugas de información ocurridas en el pasado y de análisis de las más empleadas. Sin embargo, no resulta efectivo cuando la contraseña es larga y está formada por la combinación de caracteres especiales, números, mayúsculas y minúsculas.

1.1.3 Phishing

El *phishing* es un tipo de ataque basado en ingeniería social que consiste en engañar al propio usuario para que este revele información personal como las contraseñas. Para ello el atacante suplanta un sitio web y engaña al usuario mediante el envío de correo engañoso para que acceda e introduzca sus credenciales, haciéndole creer que es el sitio web real.

1.1.4 Keylogger

Un *keylogger* es un tipo de *spyware* que un usuario instala, sin darse cuenta, al descargar un archivo de internet o al acceder a un enlace. Además, este puede ser un dispositivo hardware que instale el ciberdelincuente como conector entre el ordenador y el teclado. Este es capaz de capturar las pulsaciones del teclado, las cuales quedan registradas en un archivo que posteriormente es enviado al ciberdelincuente.

Con estos ataques, cuando un ciberdelincuente obtiene la contraseña, puede llevar a cabo todo tipo de acciones. Pueden realizar compras y ocasionar gastos económicos en la cuenta de la víctima y, si además, se emplean las mismas credenciales en redes sociales y correo electrónico pueden realizar desde espionaje hasta suplantación de identidad y enviar correos en nombre del usuario así como realizar publicaciones ofensivas en las redes sociales.

1.2 Gestión de la seguridad de las contraseñas

Otro problema, centrado en la seguridad del almacenamiento de estas contraseñas, viene dado por las empresas. Hoy en día, todavía existen empresas que almacenan las contraseñas en texto plano en sus servidores, es decir, en un formato legible. Esto es un grave problema, pues aunque la contraseña sea segura, si un atacante consigue robar la base de datos de contraseñas tiene control completo de las cuentas de los usuarios afectados.

A comienzos del año 2019 se produjo la mayor filtración de contraseñas hasta el momento, la denominada "*Collection* #I" [2] en la que más de 21 millones de contraseñas han sido expuestas en una filtración que ha sido obtenida de numerosas bases de datos diferentes.

Además, el gigante de las redes sociales, Facebook, corrigió recientemente un error que causaba que las contraseñas de muchos usuarios fueran almacenadas sin seguridad alguna, siendo visibles para los empleados [3], [4]. Posteriormente a esto, surgió la noticia en la que se indicaba que los servidores de Amazon [5] almacenaban datos, como contraseñas de usuario, sin seguridad.

Esto muestra cómo grandes empresas de servicios también son susceptibles de cometer graves errores y exponer gravemente las contraseñas de sus usuarios, por lo que tras una filtración de estas, si un usuario afectado emplease la misma para otros servicios, estas cuentas corren peligro de estar en manos de ciberdelincuentes.

1.2.1 Funciones *hash*

Almacenar las contraseñas en texto claro en una base de datos no resulta seguro, puesto que estas pueden ser robadas o ser accesibles desde dentro del sistema. Por otro lado, almacenar las contraseñas cifradas puede parecer una solución alternativa de seguridad, sin embargo en caso de obtener la clave que cifra las contraseñas, se podrían descifrar.

Para evitar esto y aumentar la seguridad, se emplean las funciones *hash* o resumen, que asigna una cadena de un tamaño variable en una cadena de un tamaño determinado. De esta manera se almacena en lugar de la contraseña en claro, un resumen de esta obtenido al procesar la contraseña. Así, en el caso de ser robada la base de datos, solo tendrían acceso a los resúmenes, los cuales no se pueden descifrar (las funciones *hash* son de un solo sentido, son irreversibles).

En la actualidad, se recomienda emplear SHA-3 [6] con un resumen de 512 bits como función *hash*.

1.2.1.1 Sal

Un atacante puede almacenar un conjunto de contraseñas y calcular sus resúmenes y después comparar esta tabla, denominada tabla precalculada, con la base de datos. Para evitar este precálculo, se emplea lo que se conoce como "sal", una cadena de bytes aleatorios que se concatenan a la contraseña, y a continuación, se calcula el resumen de ambas. De esta forma el atacante no puede hacer uso de la tabla precalculada, debido a que aunque conozca la sal, la tabla que posee solo serviría para dicha sal, y puesto que por cada usuario es una sal distinta, resulta inviable tener tablas precalculadas para cada usuario.

1.2.2 Password Based Key Derivation Function (PBKDF)

A pesar que el empleo de *hashing* junto con sal es seguro, esto no evita los ataques por tarjetas gráficas o GPU (*Graphics Processing Unit*). Un ataque basado en GPU consiste en utilizar todos sus núcleos (dependiendo del modelo, puede tener hasta 4000 núcleos [7], [8]), lo cual aceleraría en gran medida un ataque de fuerza bruta respecto a un procesador (CPU), que tiene, de media, 4 núcleos.

Las funciones de derivación de clave basada en contraseña (*Password Based Key Derivation Function* – PBKDF) son funciones *hash* más lentas, utilizadas para la gestión de contraseñas. Estas permiten hacer inviables los ataques por GPU, debido a que tienen unos parámetros configurables como memoria utilizada, número de hilos y el tiempo que debe tardar. Por ello, son actualmente la opción recomendable junto con la sal.

Argon2 [9] es el PBKDF recomendado en la actualidad.

2. Justificación y objetivos

En estos tiempos donde almacenamos todo tipo de información confidencial protegidos mediante credenciales, urge la necesidad de un correcto y seguro sistema de autenticación.

Como se verá en el siguiente apartado (Estado del arte), los sistemas actuales no son perfectos. Es por ello que en este trabajo se propone un sistema de autenticación robusto, llamado ChainAuth, basado en la tecnología Blockchain. Los sistemas de autenticación deben ser el pilar principal en los servicios de red, pues todo lo que almacenan se encuentra protegido por el acceso al usuario correcto. Por esto, plantearemos un sistema que evite muchas de las vulnerabilidades o posibles causas de robo de credenciales (contraseñas débiles, ingeniería social, etc.) para ofrecer así un sistema cómodo y seguro que lidie con dichos problemas. Al final de este documento detallaremos, de manera gráfica, cómo se ve este sistema a vista de usuario.

3. Estado del arte en los sistemas de autenticación

En los últimos años se ha producido una evolución de los distintos sistemas de autenticación, ajustándose a los cambios que han surgido en la tecnología, ofreciendo mayor o menor grado de seguridad.

La autenticación es la primera fase del proceso de conexión de un usuario a los servicios de red y aplicaciones, asegurando su identidad. Generalmente, esta consta de 2 elementos, el identificador de usuario y un elemento que garantice su autenticación.

Existen numerosos métodos de autenticación, cada uno con sus características propias. En los siguientes apartados serán analizados algunos de los más empleados.

3.1 Basados en contraseña

Los sistemas de autenticación basados en contraseña son los más populares y empleados por los usuarios. Cuanto más robusta sea la contraseña, más complejidad (temporal) tendrá un atacante para conseguirla.

La información que se protege mediante contraseña es tan privada y segura como lo son las contraseñas. Para que una contraseña sea robusta y, por tanto, difícil de obtener para un atacante, esta debería cumplir una serie de características. En primer lugar, desde el punto de vista de la seguridad, lo recomendable sería que la contraseña estuviese compuesta por un mínimo de 16 caracteres (128 bits). No obstante, es cierto que, desde el punto de vista de la comodidad, los servicios recomiendan 8 caracteres: contraseñas de esta longitud pueden ser obtenidas en varias horas (dependiendo del equipo atacante), aunque estas sean complejas. En segundo lugar, la contraseña debe tener una combinación de letras (mayúsculas y minúsculas), dígitos y caracteres especiales. Además, es importante emplear

distintas contraseñas para cada servicio e ir cambiándolas periódicamente, pues utilizar la misma en distintos servicios provocaría usurpación de identidad en todas ellas en caso de ser obtenida por un atacante. Además, utilizar la misma contraseña en varios servicios provoca que un atacante solo tenga que romper la seguridad del servicio más débil (por ejemplo, un foro pequeño) para obtener la contraseña (pues es la misma) del banco.

Sin embargo, a pesar de las recomendaciones de seguridad, la mayoría de usuarios emplean la misma contraseña (por mayor comodidad, no suelen seguir las recomendaciones de seguridad, siendo fácilmente previsibles) en diferentes cuentas. No obstante, se puede emplear varios mecanismos para poseer una autenticación más fuerte. Estos mecanismos son los siguientes:

1. Algo que sabe el usuario, como es la contraseña.
2. Algo que posee el usuario, como un teléfono.
3. Algo que identifica el usuario de forma biométrica, como la huella dactilar, el reconocimiento facial, el iris o la voz.

Todos estos factores se pueden combinar, conociéndose como **autenticación de doble factor** (2FA).

El empleo de otro factor complementando a la contraseña otorga una capa extra de seguridad a los servicios y aplicaciones a los que se accede, dificultando cualquier intento de acceso indebido, pues el atacante debería conocer/poseer el segundo factor, además de la contraseña.

En la autenticación 2FA, el primer factor suele ser una contraseña generada y conocida por el usuario (debe ser robusta) mientras que el segundo suele ser un *token* que consiste en un código aleatorio generado por una aplicación externa.

Otro segundo factor empleado, o un tercero tras el anterior, es la verificación biométrica. El factor biométrico es, en principio, el más complicado de obtener puesto que si algún atacante decidiese robar, por ejemplo, la huella dactilar, debería en primer lugar obtenerla y en segundo lugar tener acceso al sistema en el cual se emplea.

3.1.1 Gestores de contraseñas

A pesar de hacer uso del doble factor de autenticación, es necesario tener una contraseña diferente y robusta para cada servicio. Para facilitar la gestión de las contraseñas se recomienda el empleo de gestores de contraseñas. Estos gestores permiten almacenar las contraseñas (junto al nombre de usuario asociado) de todos los servicios que requieran autenticación. Además, muchos gestores permiten la generación aleatoria de contraseñas seguras (cumpliendo los requisitos de una contraseña robusta), facilitando así el uso de una contraseña segura y distinta para cada servicio.

Estos gestores hacen que un usuario solo tenga que memorizar una única clave. Esta clave suele ser empleada por los gestores de contraseña para cifrar la base de datos de contraseñas del propio usuario. Dicha clave no suele almacenarse en ningún sitio y el cifrado se realiza en el cliente.

A pesar de ello, estos gestores de contraseña no son perfectos y también han sufrido fallos de seguridad. Recientemente se publicó en "*Independent Security Evaluators*" (ISE) [10] que varios gestores (de los más utilizados) eran susceptibles de recuperar secretos: en algunos la contraseña maestra mientras que en otros las propias contraseñas de los servicios.

Además, algunos gestores no cuentan con un segundo factor de autenticación, lo cual los hace más inseguros.

3.2 Basados en certificados electrónicos

Existe otra manera para poder identificar una persona o entidad de forma inequívoca en el proceso de autenticación. Para ello, se hace uso del certificado electrónico, un documento electrónico que contiene datos que identifican al usuario. Además de estos datos, tiene asociadas un par de claves pública y privada, siendo la primera accesible para cualquiera mientras que la segunda ha de mantenerse protegida.

Para garantizar que los datos que contiene el certificado son auténticos, existe una entidad llamada Autoridad Certificadora (AC), la cual se encarga de certificar estos documentos electrónicos (claves pública + datos de identificación de usuario). De esta manera, se delega la confianza en la AC.

Como podemos ver, realizar la autenticación mediante un certificado electrónico resulta más seguro que introducir las credenciales de usuario y contraseña. Además, permite garantizar la integridad pues mediante la firma digital podemos verificar que un documento ha sido firmado por el propietario del certificado, sin sufrir alteraciones.

3.2.1 Web Authentication (WebAuthn)

WebAuthn [11] es un nuevo estándar oficial, publicado por el World Wide Web Consortium (W3C), que permite la autenticación en un servicio web sin hacer uso de una contraseña, en su lugar se emplean certificados electrónicos. Para los usuarios, el acceso a los servicios se podrá realizar mediante *tokens* o datos biométricos, sustituyendo así las contraseñas.

3.3 Basados en la delegación de la autenticación

Muchos servicios no proporcionan autenticación, en su lugar, esta es delegada en un servicio de terceros. De esta forma, el servicio indica qué proveedores de identidad acepta y un usuario puede acceder al servicio autenticándose en otro distinto.

A continuación se muestran los más conocidos:

3.3.1 Open Authorization (OAuth)

OAuth [12] es un protocolo que actúa como intermediario permitiendo a un usuario autenticarse en un servicio, aunque no tenga una cuenta en él, haciendo uso de otro, el cual actuará de autenticador.

3.3.2 Security Assertion Markup Language (SAML)

El lenguaje de marcado para confirmaciones de seguridad, más conocido como SAML [13], es un estándar basado en XML que permite que los datos necesarios para la autenticación, como usuario y contraseña, sean intercambiados para realizar la autenticación y autorización. Al tratarse de un estándar, define cómo debe usarse en las aplicaciones y la estructura del documento XML.

3.3.3 Secure Shell (SSH)

SSH [14] es un protocolo que permite establecer sesiones seguras de forma remota. Este protocolo permite realizar la autenticación con usuario y contraseña así como con certificados. Además, para mayor seguridad, puede activarse un segundo factor de autenticación, como puede ser un código QR [15].

Además, permite hacer reenvío de puertos o *port forwarding* en servicios que no proporcionan autenticación.

4. Blockchain

La cadena de bloques o Blockchain, es conocida por ser la base donde se apoya el Bitcoin, la primera criptomoneda creada en 2008 bajo el pseudónimo "Satoshi Nakamoto" [16], [17].

La Blockchain consiste en un registro global, distribuido en la red, en el que se encadenan una secuencia de datos, conocidos como bloques, de manera que solo puede ser añadido un nuevo bloque a la cadena si el resto de nodos (usuarios de la red) lo aprueban, por lo que está sometido a consenso. Una vez aprobado y añadido, la Blockchain es sincronizada y actualizada en todos los nodos. Al encontrarse replicada en la red, se garantiza que no va a ser alterada malintencionadamente, y por tanto, mantiene la integridad. Por otro lado, la información que se almacena en la Blockchain no se puede borrar, pues se trata de un registro inmutable y persistente.

El contenido de cada uno de los bloques se compone de una cabecera que contiene el *hash* del bloque anterior; la fecha y hora de la creación del bloque; el resto de transacciones hasta cubrir el tamaño; y un número que se conoce como "dificultad". Esta dificultad consiste en obtener un número de 0s que tiene que cumplir el *hash* del bloque para ser validado, así, de forma estadística, se tardan 10 minutos en encontrar un resultado válido en Bitcoin. Este número de 0s determina el grado de dificultad, cuantos más 0s haya, más operaciones serán necesarias y resultara más difícil de obtener.

Cuando se consigue este *hash* correcto, el "minero" es el encargado de añadir el bloque al registro. Puesto que cada uno de los bloques está referenciando al anterior, provoca que resulte inviable modificar cualquier bloque de la cadena, ya que modificar un dato, cambiaría el *hash* de la cabecera y, por tanto, el siguiente bloque es distinto, ya que contiene el *hash* del bloque anterior, afectando al resto de

bloques. Además, para garantizar la integridad, el resto de mineros de la red están verificando continuamente cada nuevo bloque que va a ser añadido a la cadena. Sin embargo, cada vez existen más mineros y más potencia, lo cual incrementa la dificultad del minado para un usuario corriente. Por ello, existen las denominadas piscinas (*pools*), donde se juntan varios mineros para incrementar las posibilidades de generar el *hash* válido. Esto se produce bajo los sistemas de "Prueba de Trabajo" (*Proof of Work* – PoW), provocando que sea más difícil a los mineros que trabajan de forma solitaria competir contra estos grupos de mineros. Sin embargo, esto produciría el denominado ataque del 51%, donde una sola persona o grupo posee el 51% de la capacidad de cálculo. Como solución más justa, los sistemas de "Prueba de Participación" (*Proof of Stake* – PoS) proponen que la cantidad de control que un minero pueda tener en la red sea proporcional a cuanto invierte.

La Blockchain, cada vez más, está demostrando su gran potencial, y resulta interesante emplear esta tecnología en otros ámbitos, aprovechando su potencial y sus ventajas, sobre todo en aquellos servicios delicados que requieren veracidad y no ser víctimas de falsificaciones, así como en servicios centralizados por una única entidad. Por ello, este registro distribuido es empleado en diferentes campos como los siguientes:

- Sector financiero: Garantizando la seguridad en las transacciones, dotando de mayor precisión y eficiencia.
- Servicios del Estado: Llevar un control de cada paciente y sus medicamentos en hospitales y centros de salud. En cuanto a los viajes, almacenar información sobre la identificación de cada pasajero y su pasaporte. En cuanto al gobierno, llevar a cabo en las elecciones un sistema de votos transparente, evitando el fraude y garantizando el anonimato.
- Transporte: Útil para llevar un control del transporte empresarial.

- Industria musical: Permite llevar a cabo una identificación de cada cantante y sus composiciones y asegurar el *copyright* de los autores.

En este trabajo destaca la posibilidad de emplear la cadena de bloques como un proveedor de autenticación. Esto será explicado en el siguiente apartado con el sistema propuesto denominado ChainAuth.

5. ChainAuth, sistema propuesto de autenticación soportado por Blockchain

Un sistema es tan seguro como lo es su eslabón más débil. En los servicios de red que requieren autenticación, el punto más vulnerable es la contraseña, debido a que la mayoría de usuarios no cumplen las recomendaciones para crear una contraseña segura. Esto pone en peligro el servicio de red del usuario, pues si esta contraseña, destinada a proteger su servicio, es averiguada por un atacante, todo lo que protege queda comprometido. Debido a esto, surge la idea de un sistema de autenticación donde no es necesario registrarse introduciendo las credenciales que estamos acostumbrados. Este es el caso de ChainAuth, el sistema de autenticación propuesto, que utiliza la tecnología Blockchain y permite el registro y autenticación sin necesidad de introducir contraseña, gracias al empleo de la criptografía de curvas elípticas, en concreto la curva Curve25519 bajo el sistema de firma digital Ed25519 [18] debido a que es muy rápido y proporciona mayor eficiencia; junto al algoritmo de resumen SHA3-512.

Así, empleando la tecnología Blockchain conseguimos crear una tecnología basada en la identidad digital (*Blockchain based ID as a Service* – BIDaaS). Gracias a los beneficios de la cadena de bloques, ChainAuth permite realizar una correcta validación de las identidades garantizando la seguridad y la inmutabilidad de los datos.

Por ello, ChainAuth se apoya en dos grandes propiedades que todo servicio en Internet debería de cumplir:

- Integridad: Esta propiedad se encuentra respaldada por la Blockchain.
- Autenticidad: Se proporciona un certificado que verifica, de forma única, la clave pública con su propietario.

En este sistema intervienen tres grupos de actores. El primero es el proveedor del servicio de autenticación, el segundo grupo lo forman los clientes de este proveedor, conocidos como *partners*, que ofrecen servicios de red y, el tercer grupo, los usuarios de servicios de red que ofrecen los *partners* y se relacionan con el proveedor para tener una identificación digital.

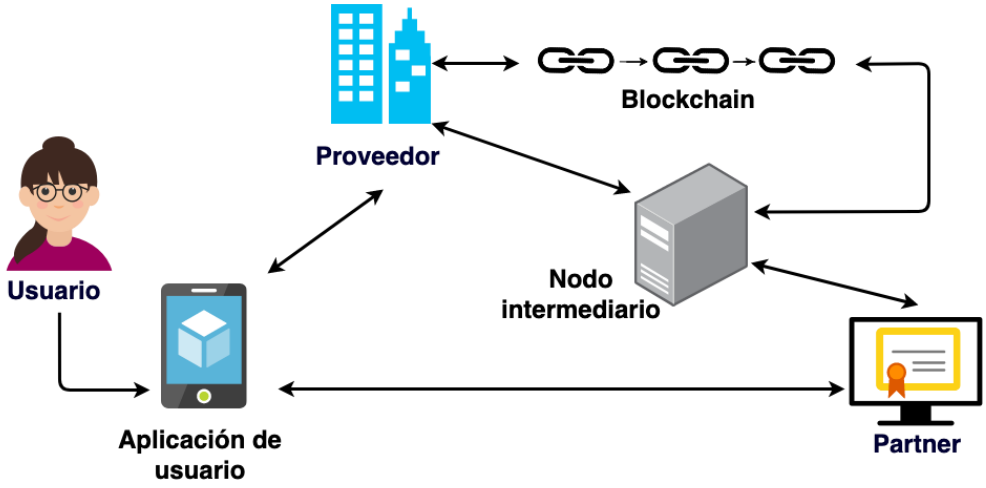

Ilustración 1: Escenario de comunicación en ChainAuth

A continuación se describen las entidades que interactúan en ChainAuth.

- Cliente (*partner*): Servicio de red externo que delega el proceso de autenticación de sus usuarios en ChainAuth.
- Proveedor: Servicio que posee la Blockchain en la que se registran los usuarios y que permite la lectura de esta a los *partners*.
- Usuario: Persona que hace uso del servicio ChainAuth para autenticarse en otros servicios de red ofrecidos por los *partners*.
- Aplicación de usuario: Aplicación móvil utilizada por el usuario para el registro en el servicio ChainAuth y autenticación en servicios externos ofrecidos por los *partners.*
- Blockchain: Blockchain que almacena en sus bloques los datos de autenticación de cada usuario.
- Nodo intermediario: Este nodo actúa como intermediario entre la Blockchain y los *partners* mediante una API (como será explicado en el apartado 5.1.2)

5.1 Descripción del sistema propuesto

Para explicar el funcionamiento completo del sistema ChainAuth, lo desglosaremos en distintos apartados, explicando cada funcionalidad del mismo. Hay que indicar que toda comunicación con el sistema ChainAuth es sobre una capa TLS (*Transport Layer Security*) en su versión 1.3 [19], sin embargo, toda la comunicación entre el *partner* y la aplicación de usuario dependerá del *partner*, aunque la aplicación soporta HTTPS y TLS. La versión 1.3, que es la más reciente, utiliza un protocolo de *forward secrecy* (cada mensaje se cifra con una clave simétrica distinta) y no permite el uso de cualquier criptosistema obsoleto o no seguro.

Nota: Algunos de los siguientes apartados son dependientes del sistema iOS para la aplicación de usuario. Para Android se plantea como futura mejora investigar si funcionaría de la misma manera.

5.1.1 Registro del usuario en el proveedor

En primer lugar, cuando un usuario quiere hacer uso de ChainAuth, debe descargarse la aplicación de usuario en su teléfono móvil y hacer uso de ella para registrarse en el servicio. En este registro, a diferencia de los servicios tradicionales donde hay que introducir el usuario y contraseña, aquí solamente debe introducir su correo electrónico, el cual debe ser seguro. No se pedirá ningún dato adicional, pues este sistema solo realiza la autenticación, facilitando la identidad digital al usuario que le permitirá autenticarse en otros servicios de red. Cualquier dato que necesite ser conocido por los clientes (*partners*) deberá ser solicitado por ellos mismos, de esta forma no se almacenan en el proveedor datos sensibles que pueda necesitar, por ejemplo, un servicio de compra, sino que solamente se almacena la información imprescindible, garantizando así el principio de mínima exposición.

Cuando un usuario *u* termina el registro; es decir, introduce su correo electrónico y lo envía mediante la aplicación de usuario al proveedor; este último

comprueba si el correo electrónico enviado ya está registrado en el sistema. Para realizar esta comprobación cuenta con una base de datos centralizada donde almacena cada correo electrónico. Si el correo electrónico ya está registrado, se informa al usuario impidiendo el uso de dicho correo. En caso de no estar registrado, el proveedor envía un correo electrónico de confirmación al correo introducido por el usuario con un *token* de sesión. Este **debe** ser abierto desde el teléfono móvil donde se está realizando el proceso de registro, puesto que ese correo electrónico redirigirá a la aplicación de usuario. Este proceso es necesario para evitar que cualquier usuario pueda registrarse con cualquier correo electrónico aunque no sea suyo. A continuación, la propia aplicación de usuario genera el par de claves pública y privada de curvas elípticas $(K_{pub}^{u}, K_{pri}^{u})$, concretamente, como se ha indicado con anterioridad, se usará Curve25519. Tras la creación del par de claves, se genera el ID virtual del usuario *u* a partir del resumen de la concatenación del email con la clave pública. Esto es $ID_{virtual}^{u} = H(email \mid K_{pub}^{u})$, donde la función *hash*, H, empleada es SHA-3 con 512 bits de resumen. Se usa esta puesto que es la función *hash* estándar actual propuesta por el Instituto Nacional de Estándares y Tecnologías (*National Institute of Standards and Technology* – NIST) [20] y 512 bits de resumen para mayor seguridad [6].

El ID virtual, el correo electrónico, la clave pública y el *token* (que confirma que dicho correo corresponde al usuario que se está registrando) son enviados al proveedor (a través de la aplicación de usuario), el cual comprobará, en primer lugar, la validez del *token* y si el ID virtual recibido es el correcto. Para ello comprueba que el *token* no esté caducado y esté asociado al correo electrónico recibido. Si es válido, procede a calcular el ID virtual con los datos recibidos del usuario *u* y así comprobar si coincide con el recibido y, por tanto, es correcto. Tras esto, el proveedor firmará el ID virtual y la clave pública de *u* con su clave privada y registrará en la Blockchain el certificado generado, junto a los datos del correo electrónico y la clave pública de *u*.

Esta firma se realiza generando un certificado de la siguiente manera: $Cert(u) = \left\{ email,\ ID^{u}_{virtual},\ K^{u}_{pub}, D^{Ed25519}_{K^{prov}_{pri}}\left[H\left(ID^{u}_{virtual} \mid K^{u}_{pub}\right)\right]\right\}$. Es decir, el certificado del usuario *u* está compuesto de: el correo electrónico; el ID virtual del usuario *u*; la clave pública del usuario *u;* y la firma con la clave privada del proveedor y el esquema de firma con curvas elípticas Ed25519 del ID virtual del usuario concatenado con su clave pública.

Una vez el proveedor ha registrado en la Blockchain al usuario, le devuelve a la aplicación de usuario el número de posición que ocupa el usuario en la cadena, que será empleado como una parte de la autenticación del usuario.

El proceso de registro de un usuario en ChainAuth se resume en el siguiente diagrama:

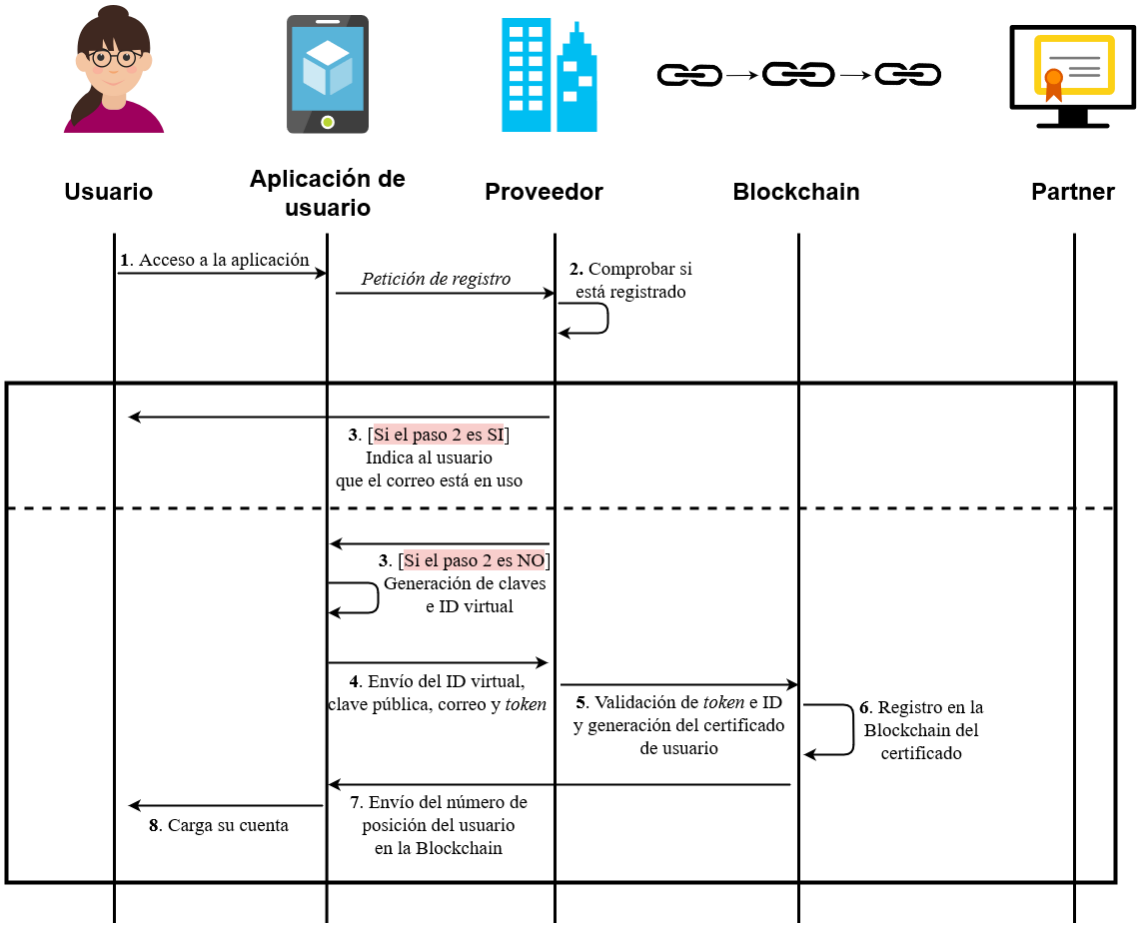

Ilustración 2: Proceso de registro de un usuario en ChainAuth

Al finalizar este proceso, se informa al usuario que debe realizar una copia de seguridad (como será explicado en los apartados 5.1.1.1 y 5.1.6.1).

5.1.1.1 Protección de la base de datos local

Mientras que la clave pública del usuario se almacena en la Blockchain, su clave privada es almacenada de forma segura en el teléfono móvil del propio usuario, en una pequeña base de datos local accesible solo desde la propia aplicación de usuario.

Al completar el proceso de registro, la aplicación de usuario genera, de forma automática y de manera segura, una clave de 256 *bits* y la almacena en el *Keychain* (o "llavero") del sistema operativo. El *keychain* es un gestor de contraseñas de Apple incluido por defecto en todos los iPhone. Este llavero tiene cifrado extremo a extremo y permite guardar su contenido en una copia de seguridad tanto en iCloud como en el ordenador personal del usuario. Esta clave es utilizada para cifrar el contenido (clave privada, ID virtual y posición de la Blockchain), de manera local, en el dispositivo del usuario. Cuando un usuario acceda a la aplicación de usuario, esta, de forma automática, recupera la clave del "llavero" y la usa para descifrar el contenido. De la misma manera, se usa para cifrar dicho contenido cuando el usuario sale de la aplicación.

El algoritmo de cifrado empleado (para cifrar el contenido que hemos comentado) es AES-256 [21] (*Advanced Encryption Standard*), puesto que es el estándar de cifrado actual y 256 como tamaño de clave al ser más seguro. El modo de cifrado empleado es GCM (*Galois Counter Mode*) [22] puesto que proporciona confidencialidad a la vez que integridad.

Al generar esta clave y guardarla en el "llavero", se le informará al usuario para que realice una copia de seguridad **cifrada** en su ordenador mediante iTunes.

De esta manera no se pierden los datos identificativos del usuario, tan solo tiene que restaurar la copia de seguridad en caso de pérdida (o cambio de teléfono móvil) y podrá seguir usando el servicio con su misma cuenta.

Como alternativa, se explica en el apartado 5.1.6.1 cómo guardar estos datos de distinta manera.

5.1.1.2 Protección de la base de datos del proveedor

Aunque la base de datos del proveedor almacene correos electrónicos y no datos confidenciales, como contraseñas, un ataque que modifique el correo electrónico puede resultar de gran gravedad para el sistema.

Si un atacante consigue acceder a la base de datos sin ser detectado, podría llevar a cabo la alteración de los correos o incluso eliminarlos. Así, si quiere suplantar la identidad de un usuario, modificaría su correspondiente correo electrónico. De esta forma, el atacante puede registrarse correctamente con el correo que ha modificado (pues este ya no existe). Finalizado este proceso de registro, puede acceder a aquellos servicios que el usuario use, como por ejemplo, redes sociales o cuentas bancarias, con la cuenta de la víctima pero las claves privada y pública del atacante. Para el servicio al que acceda, lo autentica correctamente al estar todo en orden, y el atacante puede atentar contra la privacidad de la víctima sin conocer sus claves, sino tras vulnerar la base de datos.

Para evitar este tipo de ataques y proteger a los usuarios, la base de datos debe ser protegida. Esta será replicada completamente cada 3 días, ya que solo almacena las direcciones de correo electrónico (ocupa poco espacio). Para no tener exceso de copias, estas serán eliminadas tras 30 días, de este modo nunca habrá más de 10 copias de seguridad. Haciendo esto se garantiza la disponibilidad en caso de pérdida de información o sufrir un ataque como el comentado anteriormente.

También se guardará junto al correo electrónico su correspondiente HMAC-SHA3 [23], el cual permite calcular una función *hash* junto con una clave, de manera que si no se conoce la clave no se puede recalcular el *hash* y se detectaría una modificación. En este HMAC se realiza el *hash* del correo electrónico y se emplea una clave de 512 bits introducida por el administrador. De esta forma sería inviable modificar un correo electrónico sin ser detectado, pues el atacante no podría falsificar el HMAC correspondiente. Además del HMAC, es necesario un detector de intrusos (*Intrusion Detection System* – IDS) que esté constantemente comprobando y pueda reaccionar ante cualquier ataque. De esta forma, permitirían la rápida actuación al respecto.

En caso de que este ataque, improbable (puesto que también debería conocer la contraseña del correo electrónico empleada por la víctima cuando le llegue el correo de confirmación del registro), suceda, se llevaría a cabo un proceso en el que la Blockchain impidiese autenticar la cuenta alterada, y las claves nuevas serían añadidas a la "Lista de Revocación de Certificados" (*Certificate Revocation List* – CRL) [24], siendo por tanto no válidas. Puesto que no se sabría distinguir si el correo es utilizado por un usuario legítimo o un atacante suplantándolo, el usuario afectado debe crearse un correo electrónico nuevo y seguro, y emplear una contraseña robusta y difícil de obtener para no volver a sufrir el mismo ataque, y volverse a registrar en ChainAuth.

Todas estas medidas de seguridad para evitar la suplantación se les conoce como "Defensa en profundidad", un concepto que consiste en defender todas las capas involucradas en el sistema, así dificulta que un atacante pueda vulnerar la identidad del usuario.

5.1.2 Registro del *partner* en ChainAuth

Para que un servicio de red ofrecido a los usuarios se convierta en un *partner* (cliente) del sistema de autenticación ChainAuth, primero ha de registrarse en él.

En primer lugar, se debe de definir una interfaz de programación de aplicaciones (*Application Programming Interface* – API) y su documentación adjunta para que estos potenciales *partners* lo implementen.

A continuación mostramos dicha API y sus usos (por simplicidad, se supone que todas las URLs empiezan por https://; además, todas las respuestas serán en formato JSON):

Método	URL	Variables	Funcionalidad
GET	api.*ChainAuth*.es/{$ID^{u}_{virtual}$}/p?firma=$D^{Ed25519}_{K^{ptnr}_{pri}}[H(ID^{u}_{virtual}\|p)]$		Devuelve el certificado del usuario u almacenado en la posición p de la blockchain. Sólo funciona si ya es *partner* de ChainAuth.
POST	api.*ChainAuth*.es/registro	{Certificado: {cert}, Firma: {dominio}}	Un servicio de red se registra en ChainAuth como *partner*
DELETE	api.*ChainAuth*.es/baja	{Certificado: {cert}, Firma: {dominio}}	Un servicio se da de baja como *partner* de ChainAuth.

Tabla 1: API ChainAuth

Cuando un servicio de red quiere delegar el proceso de autenticación en ChainAuth, en primer lugar realiza una petición POST según la API documentada. Esta petición contiene el certificado del servicio de red, firmado por una autoridad certificadora de confianza, y la firma de su dominio con su propia clave privada. Así, cuando el proveedor recibe el mensaje, valida el certificado y el dominio y demuestra que el servicio de red es quien dice ser. Tras esto, el proveedor registra al nuevo *partner* en una base de datos específica para los clientes de ChainAuth, añadiendo un campo que muestre el estado del *partner*, ya sea activo si está disponible y se mostrará en la aplicación de usuario o inactivo si el servicio de red se ha dado de baja en ChainAuth, el cual no será mostrado como *partner* en la aplicación. Para darse de baja del sistema, el *partner* envía una petición DELETE adjuntando su certificado y la firma de su dominio.

Al tratarse de una Blockchain privada (para acceder a ella es necesario formar parte de ChainAuth) los *partners* solo tienen acceso de lectura a través de un nodo intermediario del sistema entre la Blockchain y el *partner*. Este nodo es el encargado de atender peticiones GET por parte de los *partners* y realizar la búsqueda de usuarios en la Blockchain y devolverle los resultados.

Para demostrar que quien está haciendo uso del servicio es un *partner*, en la petición GET solicita el ID virtual y la posición (p) del usuario *u* que quiere autenticarse y el *partner* lo firma con su clave privada demostrando así la veracidad del emisor. Además, la comunicación TLS (HTTP sobre TLS) emplea autenticación basada en el cliente como una capa más de autenticado verificando que el cliente es quien dice ser.

El proceso de registro de un *partner* en el sistema es el siguiente:

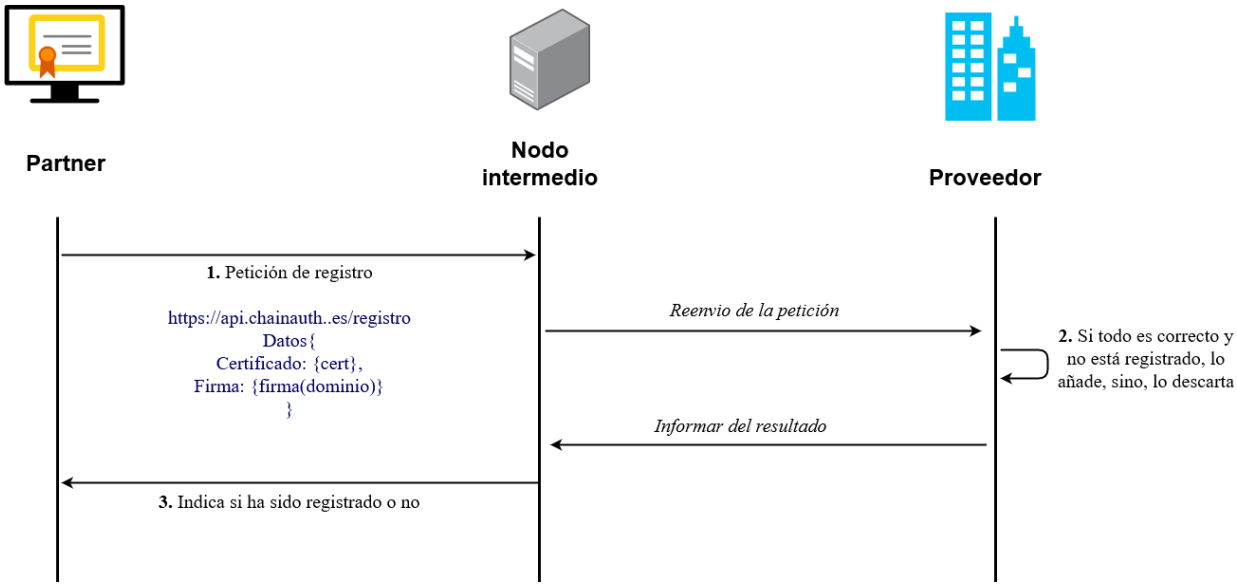

Ilustración 3: Proceso de registro de un partner en ChainAuth

5.1.3 Autenticación del usuario en los servicios

Una vez el usuario está registrado en el sistema, le aparece una pantalla nueva con todos los clientes que hacen uso de ChainAuth. A continuación, el usuario elige en qué servicio de red quiere autenticarse y comienza la interacción entre las entidades que intervienen en el proceso de autenticación: la aplicación de usuario, el *partner* y la Blockchain.

En primer lugar, tras elegir el usuario la aplicación a la que quiere acceder, se envía a través de la aplicación de usuario al *partner* un mensaje de identificación, que está compuesto por el ID virtual del usuario *u*, la fecha y hora (*timestamp*) de la petición, la posición en la Blockchain del usuario *u*, un número aleatorio único o *nonce n* y la firma de estos elementos concatenados. Este mensaje es el siguiente:

$$M = \left\{ ID^{u}_{virtual}, timestamp, posicion, n, D^{Ed25519}_{K^{u}_{pri}}[H\,(ID^{u}_{virtual} \mid timestamp \mid posicion \mid n)] \right\}$$

Se emplea un *timestamp* para evitar que un atacante realice un ataque de reproducción, en el cual captura el paquete y lo vuelve a reproducir más tarde, suplantando la identidad del usuario *u*. Además, como ya se ha comentado, la seguridad de la comunicación depende del partner, no obstante, la aplicación de usuario solo acepta HTTPS y TLS, por lo que se garantiza la seguridad a pesar de la comunicación que establezca el *partner*.

El partner, al recibir este mensaje *M* de petición de identificación, pregunta al nodo intermediario cuál es el certificado asociado al ID virtual en la posición indicada en el mensaje *M*.

El nodo intermediario, en primer lugar, comprueba que el mensaje recibido del *partner* es realmente de un partner registrado en ChainAuth. Esto lo hace preguntando al proveedor por la clave pública del dominio de donde ha venido la petición y verificando la firma adjunta del mensaje (el formato del mensaje que envía el partner al nodo intermediario fue explicado en la tabla 1 del apartado 5.1.2).

Si es un partner, procede a comprobar si el ID virtual está en la CRL o no. En caso de no estarlo, accede a la blockchain, en la posición indicada por el partner y, si existe ahí el certificado del ID virtual por el que se pregunta, se le devolverá al partner, de lo contrario, se le informará que ha ocurrido un error.

El partner, una vez tiene el certificado del usuario u, lo valida con la clave pública del proveedor. Si es correcto, obtiene la clave pública del usuario u y tras comprobar que la firma del mensaje M es correcta, comprueba si el timestamp está dentro del rango aceptado de 15 segundos de tiempo, de lo contrario se descartaría. Tras realizar todas estas comprobaciones, el *partner* permite la autenticación en su servicio ofrecido para el usuario u, el cual le aparecerá la pantalla del servicio ya autenticado. Todo este proceso se realiza de forma automática y transparente para el usuario, quien solo ve cómo selecciona el servicio a utilizar y en unos segundos se le redirige a la aplicación instalada correspondiente al servicio o al navegador predeterminado del dispositivo móvil (en caso de no tener instalada la aplicación oficial), ya autenticado.

Gracias a la Blockchain, el *partner* puede verificar de forma única al usuario, sabiendo que es quien dice ser puesto que la autenticación se basa en la clave privada del usuario que solo él posee.

En el siguiente diagrama se muestra el proceso de autenticación de un usuario:

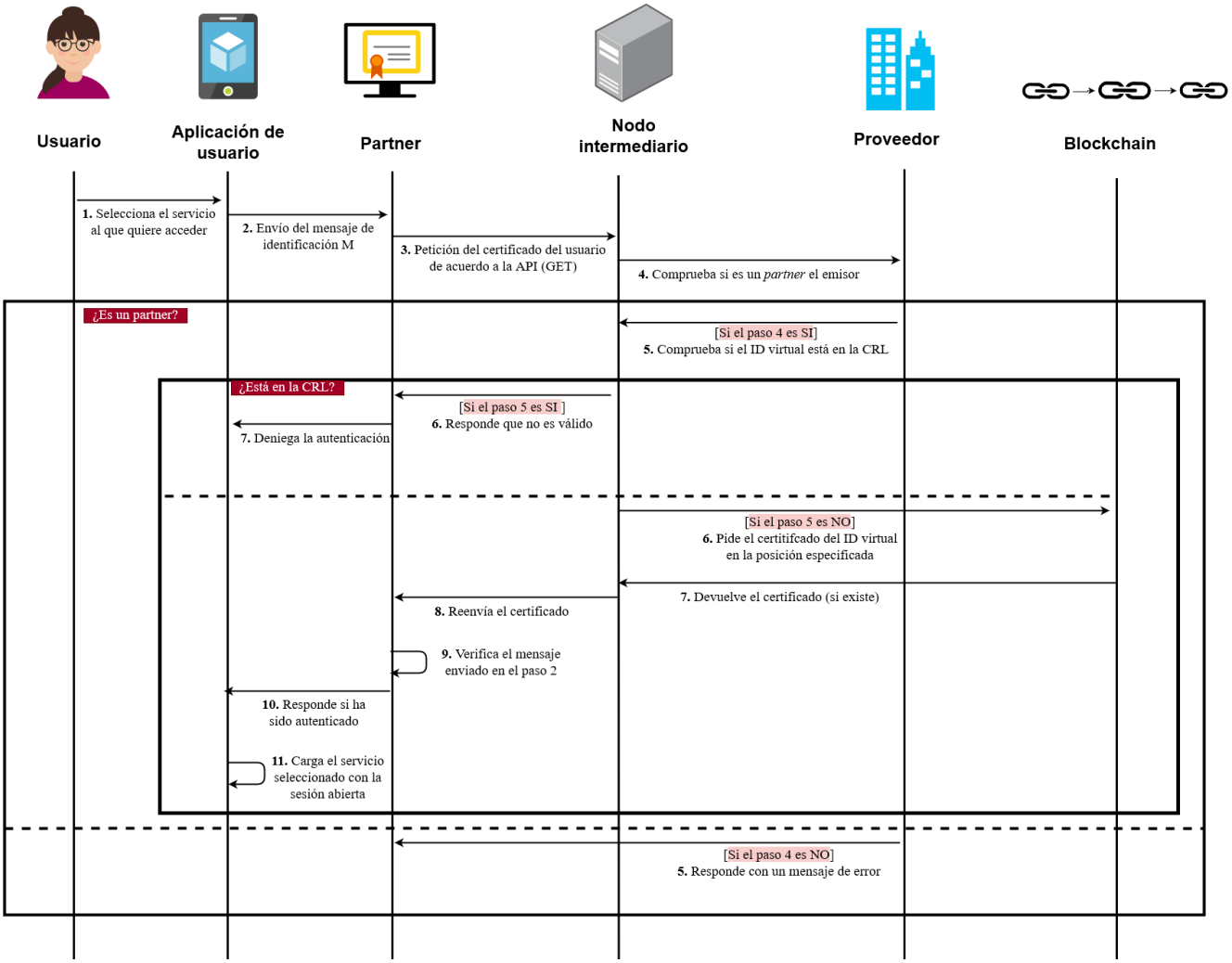

Ilustración 4: Proceso de autenticación de un usuario

5.1.3.1 Autenticación en otros dispositivos

Mientras que lo anterior es el proceso de autenticación mediante la aplicación de usuario, también puede el usuario iniciar sesión en otro dispositivo distinto, como un ordenador.

Si el usuario quiere iniciar sesión en un servicio de un *partner* desde otro dispositivo distinto a la aplicación de usuario (móvil), puede realizarlo sin tener que instalar nada en dicho dispositivo. Por ejemplo, si el usuario accede a la página web de un servicio, el cual es cliente de ChainAuth, podrá acceder a él sin necesidad de introducir sus credenciales. Para ello, habrá una opción que permita la autenticación a través de ChainAuth. Al seleccionar esta opción, el servidor del servicio web generará un *token* aleatorio pero único y lo almacenará en una base de datos local junto a un *timestamp* y a qué sesión web corresponde; y se lo manda al cliente web, el cual lo renderiza para mostrar al usuario u un código QR. El usuario u, desde la aplicación móvil de usuario escanea el *token* y comienza el proceso de autenticación en el servicio de forma transparente para el usuario.

En primer lugar, la aplicación de usuario manda una petición de autenticación al servicio añadiendo al mensaje un campo *token*. El mensaje enviado es el siguiente:

$$M = \left\{ \begin{matrix} ID^{u}_{virtual}, timestamp, posicion, n, token, \\ D^{Ed25519}_{K^{u}_{pri}}[H\ (ID^{u}_{virtual} \mid timestamp \mid posicion \mid n \mid token)] \end{matrix} \right\}$$

Cuando lo recibe el servidor del servicio, realiza la autenticación del usuario u comprobando en la Blockchain su posición e ID virtual, y si es válido el mensaje, permite la autenticación y responde a la aplicación de usuario móvil con un mensaje indicando la correcta autenticación. En cuanto al cliente del servicio web, responde con los datos de sesión, permitiendo al usuario el acceso a su cuenta. Una vez hecho esto, elimina el *token* de la base de datos temporal.

A vista del usuario, la autenticación se ha realizado escaneando con la aplicación móvil el código QR generado en la página web del servicio y, posteriormente, se recarga la web y accede a su servicio ya autenticado. Todo este proceso se ha realizado sin tener que instalar nada en el ordenador ni tener que introducir una contraseña. En la aplicación de usuario le aparece un mensaje indicando que está autenticado en el *partner*.

Este mismo proceso es llevado a cabo si el usuario decide iniciar sesión en otro dispositivo móvil, ya sea un teléfono móvil de otra persona o una *tablet*. Al acceder a un servicio *partner* de ChainAuth, le aparecerá la opción de iniciar sesión con ChainAuth y mediante la aplicación móvil que tiene en su teléfono móvil podrá escanear el código QR y hacer uso de su cuenta.

Como se puede observar, resalta la importancia que tiene el teléfono móvil del usuario, pues además de residir en él su clave privada, todo proceso de autenticación, ya sea en el propio teléfono móvil como en otro dispositivo se realiza mediante la aplicación móvil de usuario.

5.1.4 Eliminación de cuenta

Si un usuario decidiese eliminar su cuenta de ChainAuth, la aplicación de usuario contará con una opción para ello. Al darle a esta opción, la aplicación de usuario envía un mensaje con los datos del usuario *u* y el proveedor comienza el proceso de eliminación de cuenta.

En primer lugar, para evitar que un usuario se dedique a crear una cuenta y eliminarla repetidamente, en la base de datos del proveedor de ChainAuth se introducen dos campos que permitan llevar un control de cada cuenta. Así, se añadirá un campo de "Estado" que indique si la cuenta se encuentra activa o eliminada, y otro campo denominado "Número de veces registrado", de esta manera se llevará un contador de cuántas veces una cuenta ha sido registrada y eliminada. Si una cuenta lleva un número determinado de registros y procede a eliminarla, se mostrará al darle al botón un diálogo indicando que una vez borrada ya no podrá registrarse más veces. Si el usuario aceptase, la cuenta pasará de tener un estado activo a un estado de bloqueo en la base de datos del proveedor, y si el usuario quisiese registrarse de nuevo con dicho correo le aparecerá un mensaje de error indicando que dicha cuenta está bloqueada. De esta forma se garantiza que en la Blockchain no se añaden nuevos bloques repetitivos para un mismo usuario, pues lo ideal es un certificado por usuario.

Si es la primera vez que un usuario elimina su cuenta, se cambiará el estado de activo a disponible para dicha cuenta en la base de datos, así si dentro de un tiempo el usuario desea crearse de nuevo la cuenta podrá hacerlo, ya que se mostrará disponible, pero se indicará que lleva ya 1 registro realizado.

En segundo lugar, la clave pública del usuario *u* es añadida a la CRL, de manera que queda registrado como un certificado no válido. Finalmente, la aplicación cierra la sesión del usuario.

5.1.4.1 Pérdida de dispositivo móvil

Por otro lado, puede darse el caso en el que el usuario pierda su dispositivo móvil o este sea robado y, por tanto, su identidad digital, pues ahí reside su clave privada y es donde se envía el mensaje de autenticación a los *partners*. Esto es un serio problema que debe ser notificado por parte del usuario a ChainAuth cuanto antes, para evitar que otra persona haga uso de sus cuentas de forma maliciosa.

Se recomienda al usuario iniciar sesión en otro dispositivo (ya sea restaurando una copia de seguridad o importando una cuenta como se explicará en el apartado 5.1.6.1) y seleccionando la opción de "eliminar cuenta", explicada en el apartado anterior.

5.1.5 Protección de la aplicación de usuario

Cuando un usuario es registrado en la aplicación, se le muestran todos los servicios disponibles para autenticarse. Para evitar que un tercero haga uso de esta aplicación y se autentique en los servicios del usuario *u*, la aplicación de usuario debe estar protegida.

Para garantizar la seguridad de la aplicación, se implementará factores adicionales de acceso. Se podrá emplear, para asegurar el acceso a la aplicación, un *passcode* así como factores biométricos, ya sea huella dactilar o por reconocimiento facial (si lo permite el dispositivo móvil).

El uso de estos factores (*passcode* + biometría) agregaría una o varias capas de seguridad (dependiendo de cuantos factores active el usuario) y mantendría los servicios fuera del alcance de posibles atacantes, garantizando así que las cuentas del usuario no son accesibles por nadie que no sea el usuario.

5.1.6 Otras funcionalidades

5.1.6.1 Importar y exportar datos

Si por algún motivo, el usuario decidiese exportar sus datos (clave privada, posición en Blockchain e ID virtual) ya sea para iniciar sesión en otro dispositivo sin hacer uso de iTunes y tener que restaurar la copia de seguridad, o simplemente porque no quiere usar iTunes como medio de copia de seguridad, se le permitirá exportar dichos datos. Estos datos serán convertidos a un formato JSON y, posteriormente, se le pedirá una clave al usuario con la que cifrarlos. Solo se permitirá exportar los datos cifrándolos, no se permitirá hacerlo en texto claro. Esta clave será expandida a 256 bits haciendo uso de SHA-3 con 256 bits de resumen y se aplicará el algoritmo AES-256 en modo GCM (explicado en el apartado 5.1.1.1).

Una vez hecho esto, se le preguntará al usuario si quiere exportarlos vía correo electrónico o haciendo uso de la aplicación de gestión de ficheros (MEGA, Dropbox, Google Drive, etc.) que tenga instalada en el dispositivo móvil, si acaso tuviese una.

A su vez, se le permitirá la opción de iniciar sesión mediante la importación de un fichero siguiendo el formato anteriormente descrito. El usuario solo tendría que elegir desde qué aplicación importarlo y con qué clave descifrarlo.

5.1.6.2 Cerrar sesión

Al cerrar sesión solo se deben eliminar los datos locales (clave privada, ID virtual y posición en Blockchain). Por lo tanto, estos datos son sobrescritos varias veces para evitar que puedan ser recuperados.

Antes de cerrar la sesión, se le pedirá al usuario una contraseña para exportar los datos, de lo contrario no podrá recuperar su cuenta. El usuario decidirá si descartar (porque ya tiene los datos exportados) o introducir una contraseña y exportar sus datos. Tras esto se cerrará la sesión.

5.2 Ejemplos de uso práctico

En este apartado se explicarán dos ejemplos de uso práctico, el primero realiza la autenticación desde un teléfono móvil y el segundo mediante otro dispositivo, en concreto un ordenador. De esta forma, se realiza una explicación gráfica de cómo funciona ChainAuth desde la perspectiva de un usuario.

5.2.1 Autenticación desde un teléfono móvil

Cuando un usuario se descarga la aplicación de usuario y accede a ella, le aparece una pantalla principal que muestra la opción de registrarse o acceder al servicio iniciando sesión. Al final de la pantalla existe un botón de "Información" donde se explica todo sobre ChainAuth, como el modo de empleo, entre otros.

Si el usuario se registra por primera vez, le pedirá su correo electrónico, que debe ser seguro. A continuación se muestra la pantalla principal y la pantalla de registro en el sistema:

Ilustración 5: Pantalla principal de la aplicación de usuario

Ilustración 6: Pantalla de registro de usuario

El usuario, tras introducir su correo, deberá confirmar el correo electrónico que le envíe el sistema a su cuenta introducida para verificar que es él. Tras confirmar el correo, se le recarga la pantalla y se le muestra su cuenta con los servicios que utiliza. Puesto que el usuario acaba de crearse la cuenta, no le aparecerá ningún servicio agregado (como se muestra en la ilustración 7), por lo que deberá acceder a todos los servicios que son *partners* de ChainAuth, mediante el botón "Añadir servicio", y agregar a su menú cuáles son los que emplea de entre todos los que son *partners*.

Ilustración 7: Primer acceso a la cuenta en ChainAuth

Ilustración 8: Cuenta del usuario con sus servicios

Una vez el usuario ha añadido los servicios que utiliza y quiera acceder, por ejemplo, a su campus virtual, seleccionará dicho servicio. Tras quedarse cargando la pantalla, se le abrirá la aplicación (si la tiene instalada) con su cuenta ya autenticada o se abrirá su cuenta ya autenticada en el navegador web que tenga por defecto el teléfono móvil.

Ilustración 9: Servicio autenticado a través de la aplicación de usuario

5.2.2 Autenticación desde otro dispositivo

Si el usuario quiere acceder a un servicio, por ejemplo, desde su ordenador personal, para autenticarse en él necesita la aplicación móvil de usuario para escanear el código QR que mostrará el navegador web.

En primer lugar, el usuario accede desde su ordenador personal al servicio que desea acceder. En cada servicio que acceda que sea *partner* de ChainAuth se mostrará, en el apartado de autenticación, la posibilidad de autenticarse mediante ChainAuth. En la siguiente imagen, el usuario accede a Twitter y, este, al ser *partner*, le muestra dos formas distintas de realizar la autenticación, de manera tradicional mediante usuario y contraseña o haciendo uso de ChainAuth:

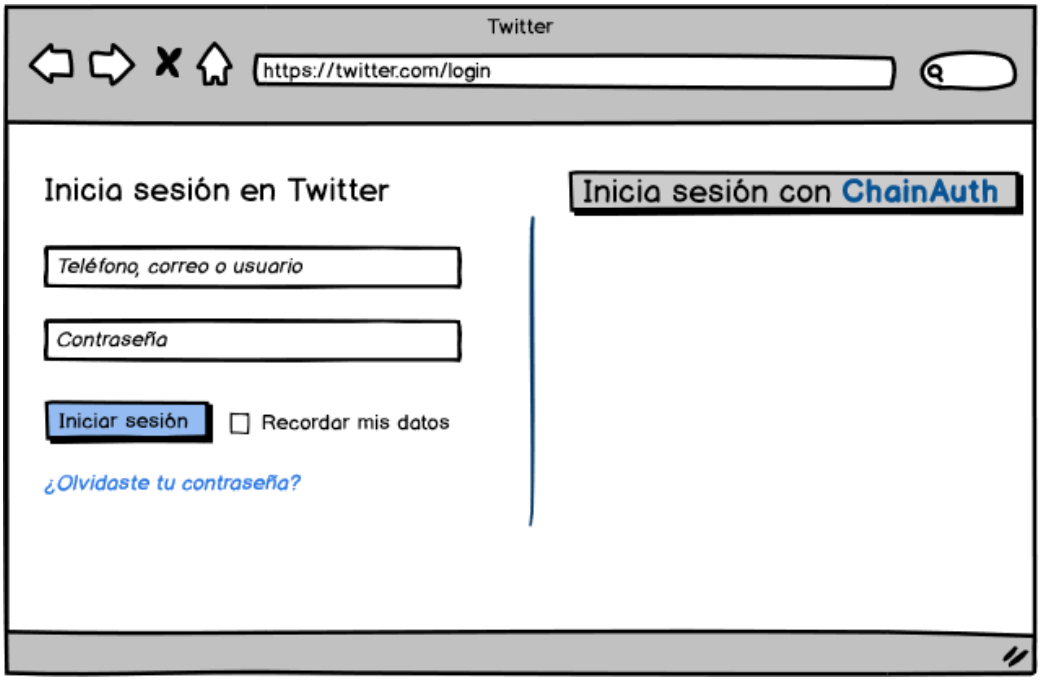

Ilustración 10: Autenticación desde otro dispositivo: ordenador personal

Al pulsar sobre el botón "Inicia sesión con ChainAuth" se generará un código QR que el usuario deberá escanear con su aplicación de usuario. Para escanearlo, el usuario debe acceder a su cuenta en la aplicación de usuario de ChainAuth y seleccionar el icono de código QR que aparece en la parte superior derecha

(como se muestra en las ilustraciones 7 y 8). Al seleccionarlo, mostrará una nueva pantalla donde podrá escanear dicho código generado en el navegador web.

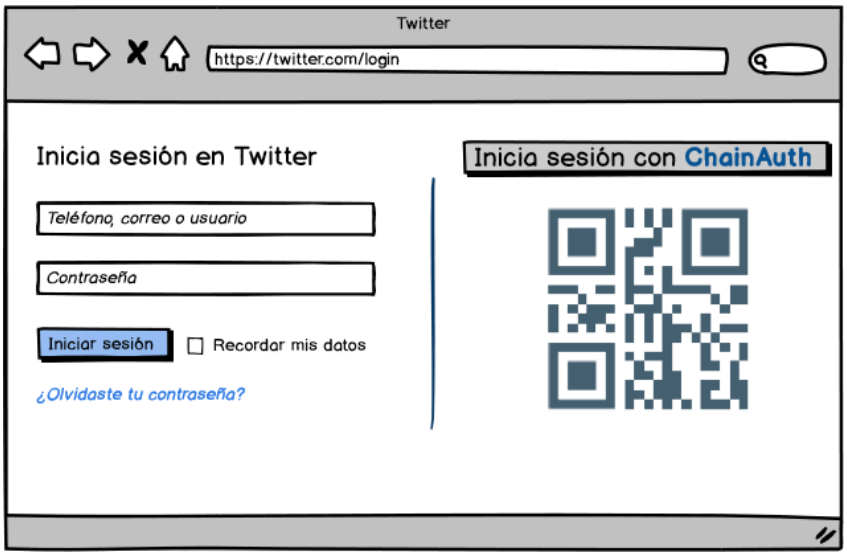

Ilustración 11: Autenticación desde ordenador personal mediante código QR

Ilustración 12: Escaneo de código QR mediante la aplicación de usuario

Una vez el usuario accede a la aplicación y escanea el código QR, el proceso de autenticación se realiza (como se explicó en el apartado 5.1.3.1) y accede de forma correcta desde el ordenador personal. En el teléfono móvil le aparecera un mensaje indicando que la autenticación ha sido realizada correctamente en el servicio (*partner*).

5.3 Sumario de funcionalidades de ChainAuth

El sistema propuesto de autenticación robusta, como hemos visto, permite mediante una aplicación móvil realizar el registro en ChainAuth, introduciendo solamente un correo electrónico, el cual debe ser de servicios respetados y centrados en la seguridad y privacidad de sus usuarios. Una vez registrado, puede autenticarse en los *partners* de ChainAuth desde el teléfono móvil o desde cualquier otro dispositivo. También permite iniciar sesión en el sistema ChainAuth reestableciendo la copia de seguridad o *backup* que hayan realizado de su cuenta o mediante la importación de un fichero cifrado con los datos. Una vez dentro de ChainAuth, el usuario puede autenticarse en servicios externos sin necesidad de introducir contraseña o en otros dispositivos mediante un código QR.

Por otro lado, el proveedor de ChainAuth cuenta con una base de datos con 4 campos que permite garantizar la seguridad y la integridad de cada usuario en el sistema. Estos campos son: el correo electrónico del usuario, su correspondiente HMAC para garantizar la autenticación e integridad de dicho correo, un campo que indique el estado de la cuenta y un campo que permita registrar el número de veces que una cuenta ha sido creada. Del mismo modo, cuenta con una base de datos de clientes de ChainAuth, en la cual indica el servicio cliente y el estado, que puede ser activo en el sistema o inactivo en caso de darse de baja.

En cuanto al acceso a la Blockchain, cuando un *partner* se registra y hace uso de ChainAuth para autenticar a sus usuarios, este tiene acceso a un nodo intermedio del sistema, que le permite realizar las peticiones de lectura en la Blockchain. Así, esta solo es accesible desde el nodo que hace de intermediario entre la Blockchain y el *partner*. Este nodo además posee la lista CRL para comprobar, cuando recibe una petición GET de un *partner* sobre un usuario, si el certificado de este usuario es válido o no.

5.4 Ventajas e inconvenientes de ChainAuth

ChainAuth es una alternativa para realizar la autenticación de los servicios. A continuación se detallan sus ventajas:

La ventaja principal consiste en **dejar atrás las contraseñas** y, por tanto, evitar que los servicios tengan todas las contraseñas almacenadas en una base de datos centralizada o que los usuarios utilicen la misma contraseña para distintos servicios. Con los servicios de autenticación actuales, si se compromete un servidor, se filtran las credenciales de todos los usuarios, en cambio, con ChainAuth, habría que comprometer usuario a usuario para robar sus claves (pues estas solo residen en sus dispositivos móviles). Además, el hecho de que el proveedor y la Blockchain no reciban ningún dato secreto beneficia la seguridad y privacidad de los usuarios. De esta forma, aunque se robe la base de datos del proveedor, no hay ningún dato comprometido, lo que provoca que para los atacantes las bases de datos no resulten útiles. De esta manera, al no almacenarse datos sensibles se garantiza el principio de mínima exposición.

Por otro lado, el hecho de que los usuarios no necesiten introducir ninguna contraseña **hace inviables los ataques de *phishing*** para obtener las credenciales de los servicios de los usuarios. Además, el no tener que pensar una contraseña para cada servicio que cumpla las características para ser considerada segura, hace que sea **más cómodo** para los usuarios el uso de ChainAuth.

Otra ventaja importante es la **seguridad en la autenticación**. Para romper el sistema y vulnerarlo, habría que romper el algoritmo de curvas elípticas empleado (Curve25519), pues el sistema de autenticación hace uso de la clave privada de cada usuario y obtenerla es, en la actualidad, computacionalmente inviable.

En cuanto a los partners, les resulta beneficioso **delegar el proceso de autenticación** en un sistema que se encarga exclusivamente de realizar una autenticación segura. Así ellos ahorran dedicar recursos para mantener esa base de datos, así como actualizar el proceso de autenticación y evitar cometer vulnerabilidades.

Sin embargo, aunque pocas, ChainAuth tiene las siguientes desventajas:

Puede ocurrir que el nodo intermedio o cualquier nodo crítico de ChainAuth caiga por cualquier motivo y nadie pueda hacer uso del sistema y autenticarse. Sin embargo, ChainAuth cuenta con numerosas medidas para que esto no ocurra y, en caso de ocurrir, se pueda restaurar el sistema lo antes posible. Estas medidas son: tener varios proveedores de acceso a Internet para que en caso de fallo de una, pueda seguir el acceso con la otra compañía; el hecho de que la base de datos de certificados sea una blockchain, ya implica de manera automática que la base de datos esté replicada; tener varios nodos intermedios para evitar que un único nodo sea el encargado de atender las peticiones y se sature así como balancear la carga. De esta forma se reduce el riesgo de que pueda dejar de funcionar ChainAuth.

Por otro lado, si el usuario no realiza copias de seguridad ni exporta sus datos y, por ejemplo, pierde el móvil, sería imposible recuperar su cuenta en el sistema. Para ello, el propio sistema advertirá al usuario de que siga unas buenas prácticas para en caso de pérdida, pueda restaurar sus datos.

Finalmente, al no tener una contraseña de acceso que solamente el usuario conozca, no hay modo de derivar una clave que sirva para cifrar los datos de cualquier servicio que utilice el usuario en el cliente. Así sería solo conocida por el cliente y el servidor seria de conocimiento cero.

6. Conclusiones

Una vez realizado este proyecto, observamos la importancia de la seguridad en el proceso de autenticación pues es el primer punto de entrada para acceder a un servicio.

En los servicios de Internet, una mala práctica de seguridad, como tener una contraseña simple, provoca que todos nuestros servicios sean vulnerables ante un atacante que intente averiguar la contraseña y, por lo tanto, tenga control total del servicio que protege.

A pesar de no existir la seguridad total, tanto los servicios de red como las aplicaciones móviles deben ser reforzadas en cuanto a seguridad. Esto es lo que hace ChainAuth, un sistema encargado de realizar la autenticación de forma segura para sus *partners*, dejando atrás las contraseñas y los problemas que ocasionan. El objetivo de ChainAuth es conseguir un sistema fiable y seguro, que sea cómodo de usar para todos los usuarios y sencillo de implementar para sus clientes. Ya no es necesario que los usuarios tengan que recordar distintas contraseñas robustas para diferentes servicios ni tampoco estar preocupados si sus contraseñas han sido filtradas porque el servicio empleado ha sido comprometido. Además, gracias al empleo de autenticación mediante certificados se consigue evitar ataques de *phishing*, muy populares hoy en día, pues ya no hay ninguna contraseña que obtener. Destaca también el empleo de la tecnología Blockchain, pues su inmutabilidad evita que los datos que almacena sean alterados.

Todo esto permite conseguir un sistema seguro basado en Blockchain que ofrece a los usuarios una manera cómoda y segura de identificarse digitalmente.

La seguridad *online* es un proceso constante que hay que estar vigilando y reforzando para garantizar la seguridad y la privacidad de los usuarios que confían en los servicios. Por ello, en ChainAuth se reduce al mínimo la superficie de ataque que pudiese agrietar el sistema y abrir paso a un posible ataque.

7. Referencias

[1] Oficina de Seguridad del Internauta, «¿Sabías que el 90% de las contraseñas son vulnerables?,» 6 Febrero 2019. [En línea]. Available: https://www.osi.es/es/actualidad/blog/2019/02/06/sabias-que-el-90-de-las-contrasenas-son-vulnerables.

[2] T. Hunt, «The 773 Million Record "Collection #1" Data Breach,» 17 Enero 2019. [En línea]. Available: https://www.troyhunt.com/the-773-million-record-collection-1-data-reach/#comment-4289914828.

[3] Facebook, «Keeping Passwords Secure,» 21 Marzo 2019. [En línea]. Available: https://newsroom.fb.com/news/2019/03/keeping-passwords-secure/.

[4] B. Krebs, «Facebook Stored Hundreds of Millions of User Passwords in Plain Text for Years,» 21 Marzo 2019. [En línea]. Available: https://krebsonsecurity.com/2019/03/facebook-stored-hundreds-of-millions-of-user-passwords-in-plain-text-for-years/.

[5] M. Kumar, «540 Million Facebook User Records Found On Unprotected Amazon Servers,» 3 Abril 2019. [En línea]. Available: https://thehackernews.com/2019/04/facebook-app-database.html.

[6] NIST, «SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions,» Agosto 2015. [En línea]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf.

[7] Nvidia, «Nvidia,» [En línea]. Available: https://www.nvidia.com.

[8] AMD, «AMD,» [En línea]. Available: https://www.amd.com.

[9] A. Biryukov, D. Dinu y D. Khovratovich, «Argon2,» [En línea]. Available: https://password-hashing.net/submissions/specs/Argon-v3.pdf.

[10] Independent Security Evaluators, «Password Managers: Under the Hood of Secrets Management,» 19 Febrero 2019. [En línea]. Available: https://www.securityevaluators.com/casestudies/password-manager-hacking/.

[11] W3C, «Web Authentication: An API for accessing Public Key Credentials,» 4 Marzo 2019. [En línea]. Available: https://www.w3.org/TR/2019/REC-webauthn-1-20190304/.

[12] D. Hardt, «The OAuth 2.0 Authorization Framework,» [En línea]. Available: https://tools.ietf.org/html/rfc6749.

[13] SAML, «SAML Specifications,» [En línea]. Available: http://saml.xml.org/saml-specifications.

[14] T. Yolent y C. Lonvick, «The Secure Shell (SSH) Transport Layer Protocol,» [En línea]. Available: https://tools.ietf.org/html/rfc4253.

[15] ISO, «Automatic identification and data capture techniques -- QR Code bar code symbology specification,» [En línea]. Available: https://www.iso.org/standard/62021.html.

[16] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System,» [En línea]. Available: https://bitcoin.org/bitcoin.pdf.

[17] D. Yaga, P. Mell, N. Roby y K. Scarfone, «Blockchain Technology Overview,» [En línea]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf.

[18] Daniel J. Bernstein, «Curve25519: new Diffie-Hellman speed records,» 9 Febrero 2006. [En línea]. Available: https://cr.yp.to/ecdh/curve25519-20060209.pdf.

[19] E. Rescorla, «The Transport Security Layer (TLS) Protocol Version 1.3,» Agosto 2018. [En línea]. Available: https://tools.ietf.org/html/rfc8446.

[20] NIST, «National Institute of Standards and Technology,» [En línea]. Available: https://www.nist.gov/.

[21] NIST, «Announcing the ADVANCED ENCRYPTION STANDARD (AES),» 26 Noviembre 2001. [En línea]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf.

[22] M. Dworkin, «Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,» Noviembre 2007. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf.

[23] H. Krawczyk, M. Bellare y R. Canetti, «HMAC: Keyed-Hashing for Message Authentication,» Febrero 1997. [En línea]. Available: https://tools.ietf.org/html/rfc2104.

[24] R. Housley, W. Polk, W. Ford y D. Solo, «Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,» [En línea]. Available: https://tools.ietf.org/html/rfc3280.

Glosario Acrónimos

GPU: (*Graphic Processing Unit*), unidad de procesamiento gráfica.

PBKDF: (*Password Based Key Derivation Function*), funciones de derivación de clave basada en contraseña.

PoW: (*Proof of Work*), prueba de trabajo.

PoS: (*Proof of Stake*), prueba de participación.

BIDaaS: (*Blockchain based ID as a Service*), Blockchain basada en identidad como servicio.

SHA: (*Secure Hash Algorithm*), algoritmo de *hash* seguro.

AES: (*Advanced Encryption Standard*), estándar de encriptado avanzado.

HMAC: (*Hash-based Message Authentication Code*), código de autenticación de mensajes basado en *hash*.

IDS: (*Intrusion Detection System*), sistema de detección de intrusiones.

CRL: (*Certificate Revocation List*), lista de revocación de certificados.

TLS: (*Transport Layer Security*), seguridad en la capa de transporte.

API: (*Application Programming Interface*), interfaz de programación de aplicaciones.